Federated Learning

Key Concepts and Algorithms

Ondrej Lukas

Reading Group in Data Mining and Machine Learning

January 14, 2022

Agenda

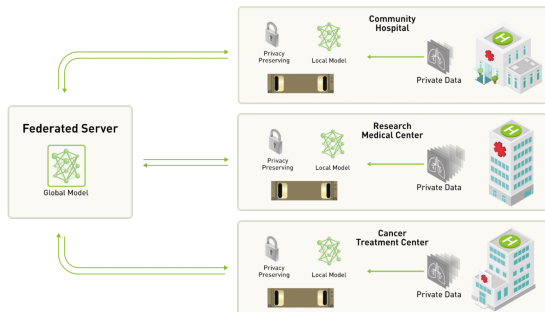
Motivation for Federated Learning

Federated Learning is a distributed machine learning approach which enables model training on a large corpus of decentralized data. **Ok, but why do we care?**

- Data is collected in the client devices
- Why not move the training there?
- Dealing with additional (new) data points
- Avoid sending, storing and processing of sensitive data in central server
- Less data for local training $\rightarrow$ lower HW requirements

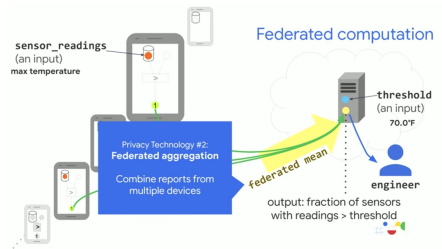
Key components

- Single master node (Federated Server)
- Multiple clients
- Data stays on the clients (privacy)
- Clients share only updates
- After aggregation, received data is discarded



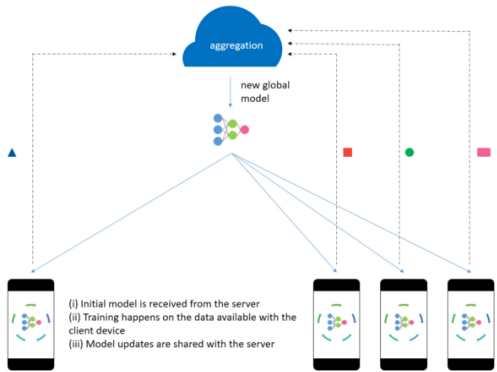
Step 1 - Federated Computation

1. Query on the server
2. Local computation per client
3. Aggregation



Step 2 - Federated Learning

- Single master node (Federated Server) with aggregated model
- Multiple clients with local models
- Training is performed in rounds
- Subset of clients can participate in each round ($C \leq 1$)



Traditional learning - recap

- For dataset D containing n samples $(x_i, y_i), 1 \leq i \leq n$ we define the training objective as:

$$min_{w \in \mathbb{R}^d} f(w) \text{ where } f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} L(x_i, y_i, w)$$

- Optimization using SGD (and it's variants) with mini-batches

$$w_{t+1} \leftarrow w_t \eta \nabla f(w_t; x_k, y_k)$$

Federated learning - training

- For dataset D containing n samples $(x_i, y_i), 1 \leq i \leq n$ distributed to K clients where P_k is the set of indices of data points on client k and $n_k = |P_k|$
- The training objective is defined as:

$$min_{w \in \mathbb{R}^d} \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

where

$$F_k(w) \stackrel{\text{def}}{=} \frac{1}{n_k} \sum_{i \in P_k} f_i w$$

Challenges of FL

- Quality of data (Not-IID)
- Unbalanced data per client
- Massively distributed
- Bottleneck in the communication (increased latency)
- Data labeling
- Easier to attack

Summary: What is different in FL?

- Data stays in the clients
- Increased latency
- Additional problems with Data
- Less computational demands

Federated SGD (FedSGD) (1)

#Samples	Learning rate	#Clients	#samples on client k
n	η	K	n_k

- A randomly selected client with n_k data samples in FL $\sim$ a randomly selected sample in the traditional learning
- FedSGD makes single step of gradient descent per round
- $C = 1$

Federated SGD (FedSGD) (2)

In round t

1. The central sever broadcasts current model w_t to each client
2. Each client computes gradient $g_k = \nabla F_k(w_t)$
3. Aggregation:
   3.1 Each client submits g_k directly to the central server which computes the update

$$w_{t+1} \leftarrow w_t - \eta \nabla f(w) = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

or

   3.2 Each client computes the the update and shares the weights $w_{t+1}^k \leftarrow w_t - \eta g_k$ and the central server computes the aggregation $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

Federated Averaging (FedAvg)

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

- Extension of FedSGD [McMahan, et al., 2017]
- Clients share updated parameters
- $C < 1$ (fraction of clients)
- $E > 1$ multiple training steps per round
- $B \leq n_k$ mini-batch in local training

Problems with FedAvg

- coordinate-wise averaging can lead to sub-optimal results
- For Non-IID data training is long and slow
- For Non-IID data averaging leads to decrease in model performance

Federated Learning with Personalization Layers (FedPer)

- Transfer learning principle in FL
- Server Aggregates only parts of the model (in green)
- Specialization Layers are biased on each client
- Aggregated model in the server is not usable on its own

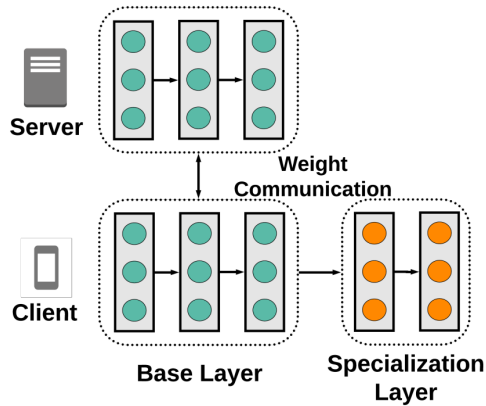


Figure: Courtesy of Ek, et al., 2021

Federated Matched Averaging (FedMA)

- Layer-wise aggregation
- Merging of similar neurons using non-parametric clustering
- Averaging of neurons within clusters
- Better performance than FedAvg
- Aggregation is very slow (`https://arxiv.org/pdf/2002.06440.pdf`)

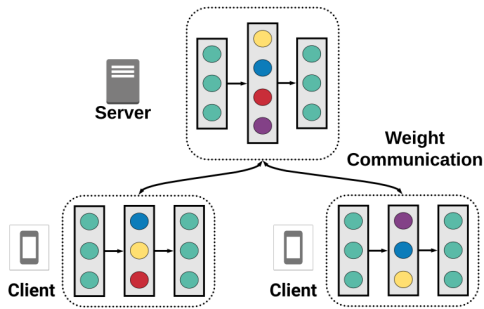
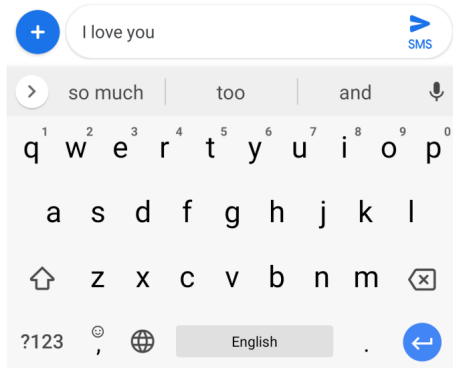


Figure: Courtesy of Ek, et al., 2021

Federated Learning for Mobile Keyboard Prediction

- Google Research [Hard, et al., 2018]
- RNN model (Coupled Input-Forget Gates) and FedAvg algorithm
- Model size: 1.4M parameters $\sim$ 1.4MB
- Each client processes $\sim$ 400 sentences per epoch
- 3K training rounds $\sim$ 600M sentences (4-5 days) until convergence



Federated Learning for Mobile Keyboard Prediction

- Google Research [Hard, et al., 2018]
- RNN model (Coupled Input-Forget Gates) and FedAvg algorithm
- Model size: 1.4M parameters $\sim$ 1.4MB
- Each client processes $\sim$ 400 sentences per epoch
- 3K training rounds $\sim$ 600M sentences (4-5 days) until convergence

Model	**Top-1 recall [%]**	**Top-3 recall [%]**
N-gram	5.24 ± 0.02	11.05 ± 0.03
Server CIFG	5.76 ± 0.03	13.63 ± 0.04
Federated CIFG	5.82 ± 0.03	13.75 ± 0.03

Table: Prediction impression recall for the server and federated CIFG models compared with the n-gram baseline, evaluated in experiments on live user traffic. [Hard, et al., 2018]

FedAvg demo

References

McMahan, Brendan, et al. (2017)
Communication-efficient learning of deep networks from decentralized data.
Artificial intelligence and statistics PMLR, 2017.

Hard, et al. (2018)
Federated learning for mobile keyboard prediction.
arXiv preprint ,arXiv:1811.03604 2018.

Questions?